Algorithmique avancée et compilation
Examen final
Sur machine, documents autorisés
Durée : 4h

On se propose de manipuler des grammaires définies comme dans l'exemple suivant[1]. L'objectif principal est de produire à partir d'un tel fichier, de la façon la plus générale possible, une grammaire "sémantique" au sens de `nltk`, telle qu'illustrée ci-après :

```
S   --> GN GV .
GN  --> DET N .
GN  --> GN GNP .
GN  --> NP .
GV  --> V .
GV  --> V GN .
DET --> la | le .
V   --> parle  .
V   --> coupe | avale .
N   --> dessert | fromage | pain .
ADJ --> petit | gentil .
```

```
%start S
S[sem=<app(?x, ?y)>]  -> GN[sem=?x]  GV[sem=?y]
GN[sem=<app(?x, ?y)>] -> DET[sem=?x]  N[sem=?y]
GN[sem=?x] -> NP[sem=?x]
...
V[sem=<\x. parle(x)>]   -> 'parle'
...
```

1. Ecrire un parser `ply` capable de reconnaître une telle grammaire et d'en construire une représentation en mémoire.
2. Ajouter au parser une analyse de la grammaire en mémoire, qui mesure les paramètres suivants :
   (a) l'axiome de la grammaire
   (b) le nombre de non-terminaux
   (c) le nombre de règles lexicales
   (d) le nombre de productions singulières
3. Vérifier aussi que le langage engendré par la grammaire est non vide[2].
4. Produire avec la grammaire parsée un fichier au format des grammaires avec attributs sémantiques `nltk`, charger cette grammaire avec les outils `nltk` (`nltk.data.load`), et vérifier que l'on peut parser avec l'analyseur nltk des phrases engendrées par cette grammaire.

   Dans un premier temps, laisser non spécifiés les calculs sémantiques[3].

   Dans un second temps, pour la détermination des calculs sémantiques, on procédera de la manière suivante :
   - Pour les règles non lexicales, on suppose que la combinaison sémantique ne dépend que du nombre de non terminaux de la partie droite de la règle.
   - Pour les règles lexicales, on suppose que l'on dispose (soit dans un fichier à part, soit dans le code lui-même) d'un ensemble de correspondances entre catégorie et λ-expression, par exemple :

     ```
     N \x. N(x)
     ADJ \x. ADJ(x)
     NP \P. P(NP)
     ```

   - On admettra que tous les déterminants ont la même valeur sémantique, celle de *un*.

Renvoyer par mail en fin d'épreuve un fichier `python`, précisément commenté, en particulier en ce qui concerne les approximations rendues nécessaire par ce sujet simplificateur.

[1] Voir le fichier `grammaire-des-dimanches.txt`

[2] On procède en général en définissant récursivement le sous-ensemble des non-terminaux ayant une contribution, c'est-à-dire qui sont à l'origine d'au moins une règle dont la partie droite ne contient que des terminaux, ou des non-terminaux ayant une contribution. Le langage engendré est non vide si l'axiome appartient à cet ensemble.

[3] Pour pouvoir utiliser `nltk.sem.text_interpret()` il faut que l'attribut `sem` soit au moins défini pour l'axiome, même avec une valeur factice (e.g., `S[sem=<x>] -> GN GV`).